Journal of the Operational Research Society (2012) 63, 2–15

www.palgrave-journals.com/jors/

Modelling and solving an *m*-location, *n*-courier, priority-based planning problem on a network

G Erdoğan[1*], B Tansel[2] and İ Akgün[2]
[1]*Ozyegin University,Üsküdar, Turkey; and* [2]*Bilkent University, Bilkent, Turkey*

In this paper, we study an *m*-location, *n*-courier, priority-based planning problem on a network, which we refer to as the Courier Planning Problem (CPP). The CPP arises on a daily basis in the context of planning the transportation of materials and personnel in peacetime for the Turkish Armed Forces. The main issue addressed in CPP is to transport as many of deliverables as possible from their origins to their destinations via a fleet of transportation assets (couriers) that operate at fixed routes and schedules. Priorities must be taken into account and constraints on the routes, operating schedules, and capacities of the transportation assets must be obeyed. Time windows may be specified for some or all transportation requests and must be satisfied. We study the CPP as well as its two extensions, and present integer programming formulations based on the multi-commodity flow structure. The formulations are tested on real world-based data and display satisfactory computational performance. Our main contributions are to develop an effective formulation scheme for a complicated large-scale real world problem and to demonstrate that such problems are solvable via commercial general purpose solvers through meticulous modelling.

Journal of the Operational Research Society (2012) **63,** 2–15. doi:10.1057/jors.2011.8
Published online 2 March 2011

Keywords: transport; optimization; military; networks and graphs

1. Introduction

In this paper we study a problem that arises on a daily basis in the context of planning the transportation of materials and personnel in peacetime for the Turkish Armed Forces. The number of transportation requests received is generally in the range of 50–100 per day. The types of commodities that need to be transported may include basic supplies, equipment, spare parts, ammunition, weapon systems, military personnel, etc. A transportation request may be associated with more than one commodity type, but these items are considered as a bundle and handled as a single item. There are priorities assigned to items based on such factors as the type of the item, the urgency required of the delivery, and the rank of the commanding officer originating the request. The items must be transported from various origins to various destinations by means of a set of transportation assets that operate at fixed routes and schedules. While the available means of transportation may include trains, trucks, boats, cargo planes, and helicopters in general, the Turkish Armed Forces prefer to use cargo planes as the principal means of transportation for reasons of security. The routes and operating schedules of the transportation assets to be used during the year are decided by each service (the army, the navy, and the air force) and remain essentially fixed throughout the year. These transportation assets are referred to as couriers by the Turkish Armed Forces. Even though the term 'courier' generally refers to a person transporting items of small size (official documents, messages, etc), it is used in this context in a broader sense to refer to a transportation asset of any kind that operates on a regular basis on a given route to transport items of any kind provided that item-carrier compatibility and capacity limits on volume, weight, and number of passengers are observed. We refer to this problem as the *Courier Planning Problem* (CPP).

1.1. The existing practice

In the existing practice, a military unit that has a need to have an item or personnel transported from its home base to a specified destination files a service request for a specific courier whose route includes stopover locations accessible to the sending and receiving military units. Service requests must give sufficient details on the priority, type, dimensions, and loading/unloading requirements of the item. Any features that require special handling must also be specified. Service requests are accepted within a designated

*Correspondence: G Erdoğan, Department of Industrial Engineering, Ozyegin University, 34662 Üsküdar, Istanbul, Turkey.
E-mail: gunes.erdogan@ozyegin.edu.tr

time window that ends a number of days prior to the scheduled departures of each courier. When the time window for receiving service requests for a given courier is closed, the service requests are sorted in decreasing order of priorities and ties are ordered in descending order of item weights. Loads are accepted starting from the top of the priority list until the capacity limit is reached. All other service requests are rejected. Owners of accepted service requests are given instructions as to when, where, and in what condition their loads must be made available for loading. Owners of rejected service requests are also notified and may file a new service request for a subsequent departure of the same courier or some other courier.

The data for transportation requests include source and destination locations, time windows for departures and arrivals, weights and volumes of items to be delivered, number of accompanying passengers (if any), and priority values. If a certain transportation asset is not suitable for the delivery of an item, this is specified as an additional restriction. The data for the service structure consist of a list of transportation assets, their types, origins, stopover points, destinations, and scheduled times of arrivals and departures en route. The transient times at visited locations en route are large enough to include times for loading, unloading, refuelling, and resting as necessary. A weight, volume, and passenger capacity is specified for each transportation asset.

In the existing practice, each service request is made to a specific courier of the sender's choice and risks rejection if competing items for the same courier fill the capacity. No transhipments are allowed between couriers. If there is no courier to deliver an item from its origin to its final destination, the sender does a preliminary study to find a way of breaking down the routing of the item into a sequence of courier routes, each associated with a specific courier, and files a separate request for each portion of the delivery in successive planning periods. Securing and storing the item between courier transfers in different planning is the responsibility of the sending unit. This situation can be improved by allowing an item to begin its journey by one of a number of alternative couriers each accessible to the military unit that sends the item. Further and more significant improvement can be achieved if transhipment between couriers is allowed in the same planning period. This permits items to begin their journeys in one courier and end in a different courier. There may be more than one courier exchange along the way. Such exchanges must be carefully planned to make sure that off-loading and on-loading be done in a well coordinated manner without causing disruptions on the courier routes and schedules. The objective is to maximize the number of requests accepted, honouring the priorities. Other variations of the problem that demand attention are the option of minimizing the total cost of the delivery without giving up on the maximum amount delivered and the option of skipping some stops if there are no items to be loaded or unloaded at those locations. This is particularly important for delivery via cargo planes to avoid unnecessary landing and take-off.

To give some perspective to the problem, the main issue addressed in CPP is to transport as many of deliverables as possible from their origins to their destinations via a fleet of transportation assets (couriers) that operate at fixed routes and schedules. Priorities must be taken into account and constraints on the routes, operating schedules, and capacities of the transportation assets must be obeyed. Time windows may be specified for some or all transportation requests and must be satisfied. We are not aware of any studies that directly deal with CPP or a similarly structured problem in the literature. In what follows, we briefly review the related studies in the literature.

1.2. Related studies in the literature

For the special case of a *single* courier, CPP reduces to a multi-dimensional 0–1 knapsack problem (Fréville, 2004) that seeks to assign as many items to the courier as possible without exceeding its volume, weight, and personnel capacities. It is well known that the feasibility form of the 0–1 knapsack problem is NP-Complete (Garey and Johnson, 1979) implying that the feasibility form of the CPP is NP-Complete even if there is a single courier with one type of capacity restriction. The case of multiple couriers is considerably more complicated. In this case, the allocation of items to couriers must be done so that the capacity restrictions on weight, volume, and personnel are honoured separately for each courier. Items must be transported from various sources to various destinations and this gives a multi-commodity flow feature to the problem. A problem in the literature that shares the multi-commodity features of CPP is the Unsplittable Flow Problem (UFP) introduced by Kleinberg (1996). The predecessors of UFP are the well known Maximum Flow Problem (Fulkerson and Dantzig, 1955; Ford and Fulkerson 1956) and the Multi-commodity Maximum Flow Problem (Grinold, 1968, 1969). Flows in predecessor problems are continuous variables whereas flows in UFP are binary variables corresponding to multiple commodities each of which can be accepted or rejected. If a commodity is accepted, then it must be routed from its source to its destination using a single path. There are profits associated with accepting each commodity and the objective is to maximize the total profit. The feasibility form of the UFP is NP-Complete as it is a generalization of the well known Maximum Disjoint Paths Problem (Karp, 1975). The UFP arises in telecommunication networks, particularly for bandwidth allocation. Due to the nature of such networks, existing studies on UFP focus on fast approximation algorithms. This approach has been studied, among others, by Kolliopoulos and Stein (1997),

Guruswami *et al* (2003), Baveja and Srinivasan (2000), and Kolman and Scheideler (2006). Baveja and Srinivasan (2000) present an integer programming formulation for the UFP that involves path variables, but have not attempted to solve it to optimality. The authors have used the linear programming relaxation of their model as a basis for an approximation algorithm. To our knowledge, no study that solves UFP to optimality has been published. CPP differs from UFP in that issues related to fixed routes and schedules of couriers, time windows, and knapsack type capacity constraints that are relevant in CPP are absent in UFP. If present, priorities in UFP can be implicitly handled by incorporating them into the profits.

A second class of problems related to CPP is the class of Vehicle Routing Problems (VRP) in which one or more vehicles departing from a depot must serve customers at various locations by visiting them once along their journeys and returning eventually to the depot. CPP differs from VRP in that the routes and schedules of vehicles are given in CPP while they are determined by the model solution in VRP. The closest relative in the VRP literature to CPP is the class of Static Pickup and Delivery Problems (PDPs) in which all customers (transportation requests) must be served by picking up commodities from their origins and delivering them to their destinations. Since all demand must be served, the concept of priority is not relevant and hence has not been incorporated. Time windows have been considered in many studies, but usually for vehicles rather than for transportation requests. We refer the reader to Berbeglia *et al* (2007) for a recent survey on PDPs.

Studies in military transportation for peacetime movement of materials and personnel consider the *Deployment Planning Problem* (DPP) rather than the CPP. DPP involves moving multiple military units as a whole (as a convoy) from its home base to its target location with all of its materials and personnel. The main issues addressed include determining a time-phased movement plan that includes routes of movement as well as an allocation of transportation assets to military units to carry out the transportation. DPP differs from CPP in major ways in its basic structure and its requirements. We refer the reader to Akgün and Tansel (2007) and Baker *et al* (1999 and 2002) for details on DPP.

As is evident from our discussion of the literature, there are more differences between CPP and other well-known problems in the literature than similarities. Our focus in the paper is to model CPP without compromising its essential features and solve it for realistic sizes. We give models that capture all important aspects of the problem and solve the main model for realistic problem sizes in low CPU times. The model with an option to skip stops has significantly more binary variables and is solved in relatively higher CPU times. We believe that the civilian sector may benefit from our approach in handling their delivery and distribution problems by utilizing a fleet of transportation assets that operate at fixed routes and schedules as opposed to frequently adjusting routes in response to changing demands.

The rest of this paper is organized as follows. In section 2, we present the core network structure on which we build the proposed model. The constraints of the problem are formulated using this structure in section 3. We construct an objective function in section 4 that properly handles the priorities. In section 5, we present variants of the basic model that consider the operating costs of the transportation assets and the option of skipping stops. In section 6, we present additional remarks on the proposed models. Section 7 consists of our computational experiments. In section 8, we give our concluding remarks.

2. Core model structure as a network

In this section, we present our modelling scheme.

2.1. Courier network

Let m be the number of couriers and n be the number of distinct locations each visited by at least one courier. Define $G=(N, A)$ to be the directed graph with node set $N=\{1, \ldots, n\}$ corresponding to the n locations serviced by the couriers and arc set A consisting of directed arcs (i,j), one for every courier whose route includes a departure from node i directly followed by an arrival at node j. Although A may contain parallel arcs (i,j) due to multiple couriers visiting vertex j after vertex i, every parallel arc is associated with a unique courier and this allows us to partition A into subsets $A_1, \ldots, A_m$, where A_i consists of those arcs used by courier i. Assign the capacity triplet ($WCap_i$, $VCap_i$, $PCap_i$) to each arc in subset A_i. The assigned triplet refers to the weight, volume, passenger capacities, respectively, of courier $i \in \{1, \ldots, m\}$. We refer to the network $G=(N, A)$ as the *courier network*. We find it convenient to let N_i be the subset of N consisting of the nodes visited by courier i and define the (sub)network $G_i=(N_i, A_i)$ to be the *courier network associated with courier i*. Note that courier networks are arc-wise disjoint but may include nodes that are jointly used by different couriers.

2.2. Time extended courier network

The courier network gives the basic transportation structure associated with item movement, but it is difficult to deal with time related issues based on this structure alone. We now define a time extended version of the courier network to account for time related issues.

Let $[0, T]$ be a time window for which the planning function is to be carried out. We assume all transportation-related activity is taking place in continuous time and that T is large enough to include the movement schedules of all couriers under consideration. A discrete subset of $[0, T]$ is extracted as follows. Define a time point $h \in [0, T]$ to be an

event time if either a departure or an arrival occurs at a node of the courier network at time point h. Event times can be directly obtained from courier schedules. Delete all time points from $[0, T]$ except the event times and let $H = \{h_1, \ldots, h_r\}$ be the resulting set with indexing done so that $h_k < h_{k+1}, k = 1, \ldots, r-1$. The *time extended courier network* $\tilde{G} = (\tilde{N}, \tilde{A})$ is defined as follows: For each node $i \in N$ and each event time $h_k \in H$, define a node with label ih_k only if a departure or arrival occurs at location i at time h_k. Let $\tilde{N}$ be the set of nodes ih_k defined in this way. Note that for each event time, there is at least one location at which an arrival or departure occur at that event time. Consequently, there is at least one node ih_k for each h_k . To define the arc set $\tilde{A}$, let (p, q) be an arc in the courier network used by courier i. Courier i visits locations p and q consecutively on its route so that there is a departure time h_k from location p and an arrival time h_l at location q . Thus, ph_k and qh_l are well defined. Connect the nodes ph_k and qh_l by a directed arc and denote this arc by (ph_k, qh_l). Let $\tilde{A}_i$ be the set of arcs formed in this way for courier i and let $\tilde{A}^1 = \cup_{i=1}^{m} \tilde{A}_i$. Define $\tilde{A}^2$ to be the set of arcs of the form (ih_a, ih_b) where ih_a and ih_b are both in $\tilde{N}$ and there is no event time h_k for which $ih_k \in \tilde{N}$ with $h_a < h_k < h_b$. We define $\tilde{A} = \tilde{A}^1 \cup \tilde{A}^2$. Each arc (ph_k, qh_l) in $\tilde{A}^1$ is associated with a unique courier i and is assigned the same capacity triplet as that assigned to arc (p, q) in A_i. Observe that $\tilde{A}^1$ has the same number of arcs as does A. The arcs in $\tilde{A}^2$ signify inactive times elapsed between an arrival and a departure of a courier at a given location and are assigned unlimited capacities. We refer to the network $\tilde{G} = (\tilde{N}, \tilde{A})$ as the *time extended courier network*. Because the couriers used in a year are decided once or twice a year, this network structure remains essentially constant during the year.

Example: Suppose we have three couriers and five locations under consideration. Courier 1 departs from location 1 at time 2, visits locations 2, 3, 4, 5, at times 3, 5, 6, 7, respectively, and returns to location 1 at time 8. We assume in this example each visit at a location is instantaneous; that is, the arrival and departure times at a location are the same. Courier 2 departs from location 4 at time 1, visits locations 3 and 1 at times 3 and 5, respectively, and returns to location 4 at time 8. Courier 3 departs from location 3 at time 1, visits locations 5 and 2 at times 4 and 6, respectively, and returns to location 3 at time 8. Figure 1 gives the corresponding courier network $G = (N, A)$. The corresponding time extended network is shown in Figure 2.

The sizes of both the courier and the time extended networks are defined by the number of arcs and nodes traversed by the couriers in service. If we assume that the route of each courier is a simple path or a simple cycle, then the courier network $G = (N, A)$ has $|A| = \sum_{i=1}^{m} |\mathrm{A}_i|$ arcs while the number of nodes, n, is at most $|A|$ since each courier route A_i has either $|A_i| - 1$ or $|A_i|$ nodes. The size of

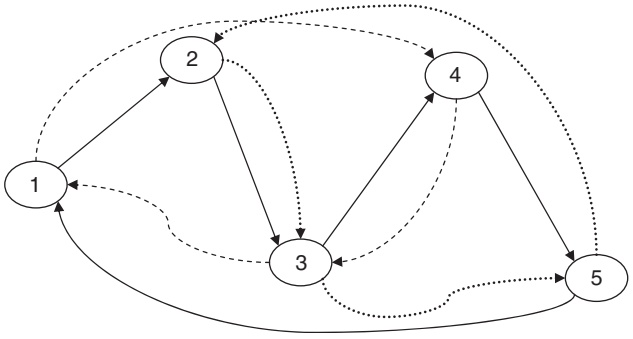

Figure 1 Example courier network $G = (N, A)$.

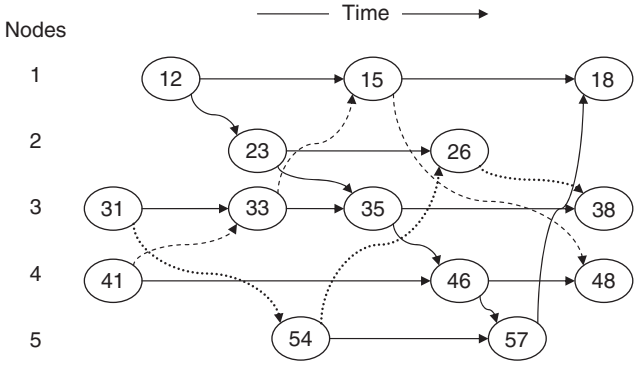

Figure 2 Time extended network $\tilde{G} = (\tilde{N}, \tilde{A})$ for the courier network of Figure 1.

the time extended network $\tilde{G} = (\tilde{N}, \tilde{A})$ is also $O(|A|)$ since $|\tilde{N}| \leqslant 2|A|$ (each arc in A gives rise to at most two time labelled nodes in $\tilde{N}$ associated with a pair of arrival and departure times) and $|\tilde{A}| = |\tilde{A}^1| + |\tilde{A}^2| \leqslant |A| + 2|A|$ since $\tilde{A}^1$ has the same number of nodes as does A and $\tilde{A}^2$ has at most $2|A|$ nodes due to the fact that each location visited by a courier may give rise to at most two time labelled nodes corresponding to an arrival and a departure from the same node. We conclude that both networks G and $\tilde{G}$ are of size $O(|A|)$. Clearly, $|A| \leqslant mn$.

The foregoing remarks on the size of the time extended courier network imply that a significant reduction in network size is achieved by constructing the network on event times as opposed to taking, for example, evenly spaced time points to approximate continuous time. The time extended network based on event times gives an exact description of all relevant activity within the planning horizon without introducing an excessive number of nodes and arcs.

3. Multi-commodity flow network and constraints

Given the network structure $\tilde{G} = (\tilde{N}, \tilde{A})$, we are able to formulate the CPP as a multi-commodity network flow

problem with knapsack type capacity constraints. Let $K=\{1, \ldots, k\}$ be the list of items for which there is a service request for transportation. For each item $k \in K$, a triplet (w_k, v_k, p_k) is given that refers to the (weight, volume, number of accompanying personnel) associated with the item. Let I_k and J_k be specified subsets of the node set N from which item k can begin and end its journey, respectively. Let $[\underline{DT}_{ki}, \overline{DT}_{ki}]$ be a time window for item k to begin its journey at an origin node $i \in I_k$ and let $[\underline{AT}_{kj}, \overline{AT}_{kj}]$ be a time window for item k to be delivered to a destination $j \in J_k$. If the time windows are invariant relative to origins or destinations, we delete the location index i and use $[\underline{DT}_k, \overline{DT}_k]$ and $[\underline{AT}_k, \overline{AT}_k]$, instead. In the event there are no time restrictions on departure or pick-up times of an item, we set the associated time windows to $[0, T]$. We create a source node s_k and a sink node t_k for each item $k \in K$ and connect s_k to t_k by a directed arc (s_k, t_k) for which the flow amount is a binary variable x_k that takes on the value 1 if item k is rejected and 0 if not. We define an external flow of 1 unit into source s_k and an external flow of 1 unit out of sink t_k.

We connect s_k to each node ih in $\tilde{N}$ if $i \in I_k$ and $h \in [\underline{DT}_{ki}, \overline{DT}_{ki}]$. Each directed arc (s_k, ih) is assigned an upper capacity of 1 unit. Letting $\tilde{I}_k$ be the set of nodes ih for which an arc (s_k, ih) is defined, we impose the constraints

$$\sum_{ih \in \tilde{I}_k} u_{k,ih} + x_k = 1, \forall k \in K \tag{1}$$

and

$$u_k \in \{0,1\}^{|\tilde{I}_k|}; \forall k \in K, x \in \{0,1\}^{|K|}, \tag{2}$$

so that whenever $x_k = 0$, item k is accepted and (1) forces some $u_{k,ih}$ to be 1 so that item k begins its journey at some node ih via a courier whose route includes ih.

A similar structure is created for the destinations by defining the directed arcs (jh, t_k) for every node jh for which $j \in J_k$ and $h \in [\underline{AT}_{kj}, \overline{AT}_{kj}]$. Assign the flow variable $v_{jh,k}$ to the arc (jh, t_k) and impose the constraints

$$\sum_{jh \in \tilde{J}_k} v_{jh,k} + x_k = 1, \forall k \in K \tag{3}$$

and

$$v_k \in \{0,1\}^{|\tilde{J}_k|}, \forall k \in K \tag{4}$$

where $\tilde{J}_k$ is the set of nodes jh for which the arc (jh, t_k) is defined.

For all arcs $a \in \tilde{A}$, define a flow variable y_{ak}, which takes on the value 1 if item k is transported along arc a via some courier i for which $a \in A_i$ and 0 if not. We impose the flow conservation constraints

$$\sum_{a \in F(ih)} y_{ak} - \sum_{a' \in B(ih)} y_{a'k} = \begin{cases} u_{k,ih} & \text{if} \quad ih \in \tilde{I}_k \\ -v_{ih,k} & \text{if} \quad ih \in \tilde{J}_k \\ 0 & \text{if} \quad ih \in \tilde{N} \backslash (\tilde{I}_k \cup \tilde{J}_k) \end{cases}, \quad \forall k \in K, ih \in \tilde{N} \tag{5}$$

and the integrality constraints

$$y \in \{0,1\}^{|\tilde{A}||K|}, \tag{6}$$

where $F(ih)$ and $B(ih)$ are the forward and backward stars of node ih, respectively. The capacity constraints for arcs are as follows:

$$\sum_{k \in K} w_k y_{ak} \leqslant WCap_a, \forall a \in \tilde{A}^1, \tag{7}$$

$$\sum_{k \in K} v_k y_{ak} \leqslant VCap_a, \forall a \in \tilde{A}^1, \tag{8}$$

$$\sum_{k \in K} p_k y_{ak} \leqslant PCap_a, \forall a \in \tilde{A}^1, \tag{9}$$

where $WCap_a$, $VCap_a$ and $PCap_a$ are the weight, volume, and passenger capacities of the courier that operates on arc a. The flow network $\hat{G} = (\hat{N}, \hat{A})$ is obtained from $\tilde{G} = (\tilde{N}, \tilde{A})$ by adding to it a source and a sink for each service request and adding arcs that connect sources to nodes in $\tilde{G}$ where a service request can begin execution and arcs that connect possible pick-up nodes to sinks. In addition, each source/sink pair is connected by a directed arc to account for possible rejects. Accordingly, $\hat{N}$ is the union of the sets $\tilde{N}$, $\{s_k : k \in K\}$, and $\{t_k : k \in K\}$, while $\tilde{A}$ is the union of the sets $\tilde{A}$, $\{(s_k, rh) : k \in K, rh \in \tilde{I}_k\}$, $\{(jh, t_k) : k \in \mathrm{K}, jh \in \tilde{J}_k\}$, and $\{(s_k, t_k) : k \in K\}$. The structure of the flow network $\hat{G}$ is depicted in Figure 3.

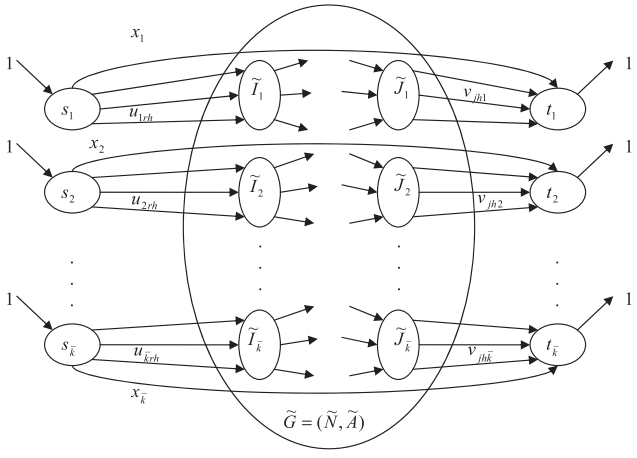

Figure 3 Structure of the flow network $\tilde{G} = (\tilde{N}, \tilde{A})$.

4. Priorities and the objective function

The main objective in the courier service system is to accept and execute as many of the service requests as possible while honouring their priorities. We assume the request for service list K is partitioned into p priority classes $S_1, \ldots, S_P$ where $p<q$ means items in priority class S_p have larger priority than those in class S_q. With an oversimplified interpretation, what this means is that it is more desirable (important) to send items in S_1 first, then items in S_2, and so on, so that one does not begin processing items in list S_k unless all items in $S_1 \cup S_1 \cup \ldots \cup S_{k-1}$ have already been accepted. This simplistic interpretation of priorities may, however, lead to a rather severe underutilization of the system. Consider, for example, a situation where priority classes $S_1, S_2, \ldots, S_k$ can all be feasibly accepted while adding any one of the items in S_{k+1} to the list of 'accepts' leads to infeasibility. Since nothing in S_{k+1} can be accepted, items in $S_{k'}, k'>k+1$ need not be considered. This of course leaves the system underutilized, as the unassigned capacity that is left after the assignment of capacities to items with priorities $1, \ldots, k$ could have been more tightly filled had a more liberal interpretation of priorities been used that would have permitted processing of items in $S_{k+2}, \ldots, S_P$ when no item in S_{k+1} qualifies for shipment together with items in priority lists $1, \ldots, k$.

To avoid system underutilization, the meaning attached to priorities is as follows: To accept a single unit of an item with priority value p, all items with priority values $q>p$ may be rejected. This perspective effectively defines a lexicographic order of multiple objectives, where the objectives are to maximize the number of transportation requests accepted for each priority value. The order of the objectives is the same as the order of their associated priority values in ascending order. To handle the priority-based multiple objectives by a single objective function, we use the scaling approach of Sherali (1982). Let n_p be the number of items in list S_p and let u_p be an objective function coefficient to be assigned to items in priority class p. The lowest priority class is $\bar{p}$ and we may set $u_{\bar{p}}$ to an arbitrary value, say, 1. We set the remaining priority values in the order $p=\bar{p}-1, \ldots, 1$ in such a way that the gain obtained from the acceptance of a single item of priority class p is more valuable than the total value that would be lost from the rejection of all items with lower priority. That is, we require $u_p > \sum_{k=p+1}^{\bar{p}} n_k u_k$ for each $p \in \{\bar{p}-1, \ldots, 1\}$. Accordingly, we choose, recursively, $u_{\bar{p}}=1$ and $u_p = \sum_{k=p+1}^{\bar{p}} n_k u_k + 1$ for $p=\bar{p}-1, \ldots, 1$ (even though many other choices that satisfy the inequalities $u_p > \sum_{k=p+1}^{\bar{p}} n_k u_k$ for $p \in \{1, \ldots, \bar{p}-1\}$ are certainly possible).

The maximization of the number of accepted requests while honouring their priorities is thus achieved by the following model:

$$\text{(P1)}\ \max \sum_{p=1}^{\bar{p}} u_p \Big(\sum_{k \in S_p} (1 - x_k) \Big) \tag{10}$$

subject to (1)−(9).

5. Cost considerations and skipping stops

Variation 1: Cost considerations

Model P1 maximizes the priority-based sum of accepted requests assuming that all couriers are in operation. Sometimes it is necessary to relax this assumption and allow some couriers to be not executed unless there is a well justified demand for their use. This may be the case, for example, with cargo planes that operate on a fixed schedule only if there is a demand for service. Knowing whether or not to operate a courier under a given set of circumstances is a non-trivial matter and it is better to let the model decide if a courier should be executed or not. In the cost-based model, we assign more value to the delivery of items than cost savings resulting from cancellation of couriers. This is accomplished by assigning a judicious choice of objective function coefficients where the delivery of the request with the least priority is more preferable than a cost saving resulting from a cancellation of courier(s). With that, the model achieves an optimal delivery of requests with the least costly activation of couriers. If disconnectedness in the network occurs due to courier cancellations, it occurs in such a way that the optimal delivery pattern proposed by the model is not disturbed. Since the transportation request data is given and is not subject to change, cancelling a courier does not cause any complications unforeseen by the model.

To this end, let c_i be the cost of executing courier i. Define F to be the set of couriers for which we have an option of not executing. For each $i \in F$, let z_i be a binary variable that is equal to 1 if courier i is executed and 0 if not. Assuming that honouring the priorities and delivering the items is more important than cost reduction, we set the lowest priority level in such a way that $u_{\bar{p}} > c_F \equiv \sum_{i \in F} c_i$. The remaining priority coefficients are set again to satisfy the inequalities $u_p > \sum_{k=p+1}^{\bar{p}} n_k u_k$ for $p \in \{1, \ldots, \bar{p}-1\}$ One legitimate choice is to take $u_{\bar{p}} = c_F + 1$, and $u_p = \sum_{k=p+1}^{\bar{p}} n_k u_k + 1$ for $p=\bar{p}-1, \ldots, 1$.

Model P2 below maximizes the total delivery, honouring priorities, while minimizing the total cost of couriers.

$$\text{(P2)}\ \max \sum_{p=1}^{\bar{p}} u_p \Big(\sum_{k \in S_p} (1 - x_k) \Big) - \sum_{i \in F} c_i z_i \tag{11}$$

subject to (1)−(9), (7′)−(9′), and (12),

where (7), (8), (9) are the same as before except that the set $\tilde{A}^1$ is replaced now by $\tilde{A}^1 \setminus F$ and (7′), (8′), (9′), and (12) are as follows:

$$\sum_{k\in K} w_k y_{ak} \leqslant VCap_a z_i, \forall a \in A_i, i \in F \qquad (7')$$

$$\sum_{k\in K} v_k y_{ak} \leqslant VCap_a z_i, \forall a \in A_i, i \in F, \qquad (8')$$

$$\sum_{k\in K} p_k y_{ak} \leqslant PCap_a z_i, \forall a \in A_i, i \in F. \qquad (9')$$

$$z \in \{0,1\}^{|F|} \qquad (12)$$

Variation 2: Skipping stops

Even though there is not much motivation for skipping stops in surface transportation, the same is not true in general for air and sometimes for sea transportation. A cargo plane that visits intermediate stops en route need not do so unless there is a need to deliver or pick-up a load at such a stop. The network model we have under consideration can easily be modified to permit couriers to skip stops whenever the economies of the situation make this a preferred alternative. If a courier i has the option of skipping stops, we simply add new arcs of the form (kh, lh') to $\tilde{A}_i$ whenever k and l are two locations visited non-consecutively by courier i with departure from k and arrival at l taking place at times h and $h' > h$, respectively. Consider, for example, courier 2 of Figure 1 whose route visits nodes 4, 3, 1, and 4 with arrival and departure times taking place at times 1, 3, 5, and 8, respectively. The resulting time extended route in Figure 2 is $41 \to 33 \to 15 \to 48$. If skipping stops is permitted for this courier, the time extended route of this courier is supplemented with new arcs $(41, 15)$, $(41, 48)$, and $(33, 48)$.

Let $\tilde{A}^1_{i,\,\mathrm{exp}}$ be the expanded arc set obtained from $\tilde{A}_i$ by adding to it new arcs, as described above, that permit courier i to skip stops. Define $\tilde{A}^1{}_{\mathrm{exp}} = \cup_{i=1}^{m} \tilde{A}_{i,\mathrm{exp}}$. Each new arc carries the same capacity triplet as that defined for the courier it is associated with. New flow variables y_{ak} are introduced for each new arc a and item $k \in K$.

The variant of P1 that permits skipping stops is essentially the same as P1 except that the set $\tilde{A}^1$ in constraints (7), (8), and (9) is replaced now by $\tilde{A}_{\mathrm{exp}}$. We also modify (6) and write $y \in \{0,1\}^{|\tilde{A}^1_{\mathrm{exp}} \cup \tilde{A}^2||K|}$.

To obtain the variant of P2 that has the option of skipping, the cost portion of the objective function needs to be modified. In P2, c_i stands for the cost of executing courier i. This cost may be taken to be the sum of arc costs associated with courier i. When a courier i is executed but does not traverse all of its arcs, we expect the courier cost to be reduced accordingly. To accommodate for this situation, let c_a be the cost of arc $a \in A_{i,\mathrm{exp}}$. Each arc cost c_a is defined relative to a unique courier and reflects accordingly the fuel and operation costs of that courier. Define the binary arc variables t_a which takes on the value 1 if arc a is used by its uniquely defined courier and 0 otherwise.

For each courier i, let $o(i)$ and $d(i)$ be the start and termination nodes (in $\tilde{N}$) for courier i. That is, $o(i) = vh$ if courier i begins its journey at a location v at time h and $d(i) = v'h'$ if this journey ends at a location v' at time h'. It is possible that $v = v'$ but $h' > h$ so that $o(i)$ and $d(i)$ are distinct. For each courier $i \notin F$, we define an external flow of 1 unit into node $o(i)$ and an external flow of 1 unit out of node $d(i)$. For each $i \in F$, the inflow and outflow at $o(i)$ and $d(i)$ are z_i (determined by the model). Define also $F_i(vh)$ to be $F(vh) \cap A_{i,\mathrm{exp}}$ and $B_i(vh)$ to be $B(vh) \cap A_{i,\mathrm{exp}}$. These are the restrictions of the forward and backward stars of node vh to arcs in $A_{i,\mathrm{exp}}$. We impose the constraints

$$\sum_{a\in F_i(vh)} t_a - \sum_{a'\in B_i(vh)} t_{a'} = \begin{cases} 1 & \text{if } vh = o(i) \text{ and } i \notin F \\ z_i & \text{if } vh = o(i) \text{ and } i \in F \\ -1 & \text{if } vh = d(i) \text{ and } i \notin F \\ -z_i & \text{if } vh = d(i) \text{ and } i \in F \\ 0 & \text{otherwise} \end{cases}, \forall vh \in \tilde{N} \text{ and } i \in \{1, \ldots, m\} \qquad (13)$$

and

$$t \in \{0,1\}^{|\tilde{A}_{\mathrm{exp}}|}. \qquad (14)$$

In addition, we replace the capacity constraints (7′), (8′), (9′) by (7″), (8″), (9″) below.

$$\sum_{k\in K} w_k y_{ak} \leqslant VCap_a z_i, \forall a \in A_i, i \in F, \qquad (7'')$$

$$\sum_{k\in K} v_k y_{ak} \leqslant VCap_a z_i, \forall a \in A_i, i \in F, \qquad (8'')$$

$$\sum_{k\in K} p_k y_{ak} \leqslant PCap_a z_i, \forall a \in A_i, i \in F. \qquad (9'')$$

Hence, the variant of P2 that permits skipping of stops is as follows:

$$(\mathrm{P3}) \max \sum_{p=1}^{\bar{p}} u_p \left(\sum_{k\in S_p} (1 - x_k) \right) - \sum_{a\in \tilde{A}^1_{\mathrm{exp}}} c_a t_a \qquad (15)$$

subject to (1)–(6), (7″)–(9″), (12)–(14).

we note that $u_{\bar{p}} = \sum_{a\in \tilde{A}_{\mathrm{exp}}} c_a + 1$ and

$$u_p = \sum_{k=p+1}^{\bar{p}} n_k u_k + 1 \text{ for } p = \bar{p} - 1, \ldots, 1.$$

6. Additional remarks

The basic model P1 and its variants include the minimum cost multi-commodity network flow problem (Bazaraa *et al*, 1990) as a special case where $w_k = v_k = p_k = 1, \forall_k$ and $WCap_a = VCap_a = PCap_a, \forall a$. In addition, the capacity constraints are knapsack constraints implying that the problems P1, P2, and P3 include the knapsack problem as a special case. It is well known that the recognition forms of the multi-commodity flow and knapsack problems are NP-Complete (Garey and Johnson 1979). Thus, the problems studied in this paper are NP-Complete.

If the time windows for an item for initial departure and final delivery are such that no combination of couriers can feasibly carry the item to one of its designated destinations, the model rejects the item (ie $x_k = 1$) regardless of the item's priority. Such infeasibilities can be deleted prior to running the model by checking to see if there exists at least one directed path in $\tilde{G}$ from origins $\tilde{I}_k$ to destinations $\tilde{J}_k$. Any path checking algorithm (Cormen *et al*, 2000) can be used to achieve this with a time bound of $O(|\tilde{I}_k||\tilde{A}|)$. If pre-processing is not done, the model detects such cases as rejects. Note, however, that the set of rejects may also include many items that can be feasibly delivered while priority (and cost) considerations may prohibit their acceptance.

For accepted items k, the flow variables u, y, v and constraints related to item k ensure that there is a path from some node in $\tilde{I}_k$ to some node in $\tilde{J}_k$ consisting of arcs for which the flow variables of item k are all 1. If this path lies on a single courier route, then the item is delivered by a single courier. If not, the item is transferred from one courier to the next one along the way as detected by courier switches at nodes where the incoming arc flow and outgoing arc flow are associated with different couriers.

For each courier i, the load content can easily be detected by studying the flow variables y_{ak} that assume the value 1 for arcs a that belong to $\tilde{A}_i$. In fact, the pick-up and delivery points of each item k carried by courier i are also detected by flows in arcs of $\tilde{A}_i$ so that a three dimensional container loading problem can be solved for courier i, based on its deliveries, that arranges items to be delivered by taking into account their pick-up and delivery points. This model's output in this sense provides a basis for optimization of more operational problems such as how to arrange the loads of the transportation asset along the way.

As final remarks, we would like state that the graph $\hat{G}$ can be reduced for instances with sparse demand and we have discovered a set of valid inequalities that are useful for pruning the branch and cut tree. We choose to skip these results for the sake of brevity and refer the interested reader to the online appendix available at http://eresearch.ozyegin.edu.tr/xmlui/handle/10679/127.

7. Computational experiments

In this section, we present our computational experiments for the cases of single and multiple couriers, using P1, P2, and P3.

7.1. The case with a single courier

We first present a computational experiment based on a single courier and five priority classes with different instances defined by different sets of transportation of requests. We use 25 generated instances to compare the performance of our model P1 (and P2) with that of the existing practice. As explained in the introduction section, the existing practice is a heuristic that consists of sorting the requests by ascending order of priorities, with ties broken in favour of larger weights, and accepting as many requests as possible in this order until no additional item can be accepted without exceeding the remaining capacity. Model P2's solution is the same as that of model P1 for the case of a single courier. The courier under consideration has one origin, one destination, and four intermediate stops. The corresponding courier graph is a simple path with six nodes and five arcs. To generate the request data, we use different values of a control parameter which we refer to as the *request density*. The request density represents the ratio of the total weight of all requests to the total carrying capacity of all couriers under consideration. Using real world data as a basis, we randomly generate 25 instances where each set of five instances corresponds to a request density of 80, 90, 100, 110, and 120%, respectively. Note that a request density of 100% generates approximately 100 transportation requests. Each request has the same time window defined by the entire planning horizon. Each generated transportation request has a single source and a single sink each on the route of the courier. The results are given in Table 1. The column labels 1, 2, 3, 4, and 5 in the table refer to the five priority classes under consideration and the entries in these columns refer to the number of accepted requests in the corresponding priority classes. The two columns labelled 'Total' give the total number of accepted requests for the two methods. The last column indicates the percent improvement achieved by the model in comparison to the existing method. The computational experiments of the 25 instances are conducted on a workstation with a 3.0 Ghz CPU and 1 GB RAM, using C++ and CPLEX 10.1. The runtime for each of the 25 runs is no more than 0.01 CPU second.

A detailed study of Table 1 reveals that the number of accepted requests in priority classes 1 and 2 are the same for both the existing method and the model solution in all of the 25 instances. For priority class 3, however, the model solution accepts a larger number of requests in 18 of the instance instances than the existing method. The difference

Table 1 Performance comparison of the existing method and the model output

Instance	*Request density (%)*	*Existing Practice*					*Total*	*Model P1*					*Total*	*Imprv. (%)*
		1	2	3	4	5		1	2	3	4	5		
1	80	9	6	12	3	5	35	9	6	13	7	4	39	11.43
2	80	5	6	15	15	8	49	5	6	15	15	10	51	4.08
3	80	10	6	12	9	6	43	10	6	13	13	5	47	9.30
4	80	4	8	23	10	2	47	4	8	23	13	5	53	12.77
5	80	3	10	16	13	9	51	3	10	16	14	11	54	5.88
6	90	5	19	14	6	2	46	5	19	16	10	5	55	19.57
7	90	11	11	12	5	9	48	11	11	13	8	8	51	6.25
8	90	3	6	17	14	4	44	3	6	17	20	6	52	18.18
9	90	6	12	18	9	6	51	6	12	20	11	6	55	7.84
10	90	5	9	21	11	4	50	5	9	21	15	5	55	10.00
11	100	9	16	18	3	4	50	9	16	23	5	5	58	16.00
12	100	8	13	14	6	3	44	8	13	17	10	4	52	18.18
13	100	7	15	16	6	5	49	7	15	19	6	8	55	12.24
14	100	5	13	17	10	9	54	5	13	19	18	14	69	27.78
15	100	4	17	18	7	8	54	4	17	20	13	9	63	16.67
16	110	6	19	12	9	3	49	6	19	13	13	7	58	18.37
17	110	9	10	19	13	11	62	9	10	20	17	12	68	9.68
18	110	7	12	16	11	3	49	7	12	20	10	6	55	12.24
19	110	6	11	16	9	3	45	6	11	21	16	2	56	24.44
20	110	8	15	18	10	4	55	8	15	18	13	15	69	25.45
21	120	7	13	11	9	7	47	7	13	19	14	10	63	34.04
22	120	11	13	9	14	9	56	11	13	14	14	11	63	12.50
23	120	6	22	9	10	7	54	6	22	14	12	7	61	12.96
24	120	9	14	21	11	15	70	9	14	24	14	14	75	7.14
25	120	8	17	8	7	7	47	8	17	12	11	7	55	17.02

between the model and the existing method in the number of accepted items in priority class 3 ranges between 1 and 5 in these 18 instances. The model accepts a larger number of requests of priority class 4 in 21 of the instances and the difference between the two methods ranges between 1 and 8 in favour of the model in those instances. For priory class 5, the model accepts a larger number of requests in 19 of the instances and the difference ranges between 1 and 11 in those instances in favour of the model. For priority classes 1, 2, and 3, there are no instances for which the existing method accepts more requests than the model. For priory class 4, there is one instance (instance 18) in which the existing method accepts more requests than the model. The difference is 1 in favour of the existing method. For priority class 5, there are five instances in which the existing method accepts more requests than the model. The difference is again 1 in each of these five instances.

The columns for the total number of accepted requests indicate that the model accepts more requests in all 25 instances than the existing method. The difference between the two methods ranges between 2 and 16 in favour of the model. For priory classes 1, 2, and 3, the accepted number of requests by the model is never less than the accepted number of requests by the existing method while, for priority classes 4 and 5 collectively, the model accepts 1 *less* number of requests than the existing method in a total of six instances and 1–14 *more* requests than the existing method in 19 of the instances. The overall improvement in the total number of requests accepted is given in the last column. The average improvement is 14.80%, with maximum improvement being 34.04%. The improvement is more pronounced in cases with a higher request density.

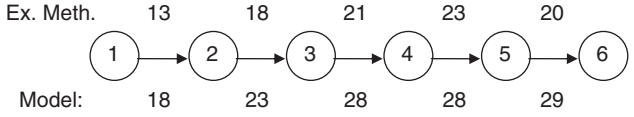

Figure 4 Comparison of solutions of the two methods for problem instance 21.

In addition we compare the outputs of the two methods based on problem instance 21. This is the instance that gives the most deviation between the performances of the two methods. Figure 4 depicts the number of requests being transported on each arc on the basis of the solutions of the existing method and of the model. The model solution carries more items in each arc than the solution of the existing method and the difference is in the range of 5–9 items. This instance has 120 transportation requests corresponding to 120% request density. A detailed analysis

of the allocation of the 120 transportation requests reveals that the model favours transportation requests that require fewer numbers of arcs, thus allowing more space for remaining requests. We may conclude that the proposed model gives solutions that better utilizes a courier's capacity than the existing method.

7.2. The case with multiple couriers

We now present our computational tests for models P2 and P3 for multiple couriers. The tests are conducted on a workstation with a 3.0 Ghz CPU and 1 GB RAM, using C++ and CPLEX 10.1.1. The models are run to optimality with default settings of CPLEX except that the relative optimality gap is set to absolute 0 instead of the relative optimality gap limit of 10^{-4} in the default setting. This is done to avoid suboptimal results, regardless of how small the deviation from optimality might be, and thus to test the performance of the proposed models under strenuous conditions. We leave out model P1 from our computational tests for the sake of brevity. The tests are performed for multiple couriers with the number of transportation requests ranging from 700 to 1100. To construct a simulated real world setting, we use the 25 most populated cities of Turkey, based on the census of 2000. We assume all transportation is done via cargo planes. The names and coordinates of the 25 cities are listed in Table 2.

Table 2 Most populated 25 cities in Turkey (census of 2000)

City no	*Name of city*	*Latitude*	*Longitude*
1	Istanbul	41.03	28.98
2	Ankara	39.92	32.85
3	Izmir	38.41	27.15
4	Bursa	40.18	29.06
5	Mersin	36.80	34.63
6	Adana	36.99	35.32
7	Gaziantep	37.08	37.40
8	Konya	37.87	32.49
9	Antalya	36.88	30.70
10	Diyarbakır	37.92	40.23
11	Kayseri	38.73	35.48
12	Eskişehir	39.78	30.52
13	Tarsus	36.92	34.90
14	Şanlıurfa	37.16	38.80
15	Malatya	38.35	38.31
16	Samsun	41.29	36.33
17	Erzurum	39.91	41.27
18	Kahramanmaraş	37.57	36.93
19	Adapazarı	40.78	30.40
20	Van	38.50	43.40
21	Denizli	37.78	29.08
22	Elazığ	38.68	39.21
23	Gebze	40.80	29.44
24	Sivas	39.75	37.02
25	Batman	37.88	41.12

We divide the cities into four different geographical zones, each to be served by a separate courier referred to as a *regional courier*. The four regional couriers have cyclic routes and operate on the first, second, third, and fourth day of each week, respectively. To connect these geographical areas, we add a fifth courier, which we refer to as the *cross-regional courier*. The cross-regional courier visits the starting and ending locations of the regional couriers as well as the four remaining most populated cities. A map detailing the routes of the couriers is given in Figure 5 and the list of stops on the route of each courier is given in Table 3. The distances between the locations are computed by scaling the result of Euclidian distance formula with the given coordinates by a factor of 92.85 km. A vehicle speed of 300 km/h and a transient time of 45 min at each stop are assumed. The couriers start their routes at 8:00 in the morning. The schedules for the rest of the stops may be computed on the basis of the data given above.

The time extended courier network for this instance can be visualized in the form of a rectangular arrangement of nodes ih_k with rows corresponding to the city indices i ($i = 1, \ldots, 25$) and columns corresponding to event times h_k ($k = 1, \ldots, 66$) such that nodes ih_k are defined for those (i,h_k) combinations for which there is an arrival or departure at time h_k at city i. A courier departing from a city p at time h_r and arriving at city q at time h_s defines an arc between rows p and q with tail at node ph_r and head at node qh_s. Horizontal arcs in the same row are defined for node pairs ih_k, ih_l to represent idle times at nodes between events taking place at those nodes. The flow network is obtained by appending as many sources and sinks to the time extended courier network as there are transportation requests and connecting each source to the corresponding sink as well as to those nodes of the time extended courier network at which the item may begin its journey. The sink nodes are connected similarly to account for possible ending locations of the item's journey. Since these networks are too large to be displayed on a single page, we skip their representation.

We use a planning horizon of 2 weeks that involves executing each regional courier twice in the planning horizon (once for each week) and executing the cross-regional courier once (in the first week) of the planning horizon. This setting represents the maximum planning horizon and the maximum number of couriers executed in the current practice, results in a total of nine couriers and 58 legs, and will be used for the rest of our experimentation. As in the previous experiment, we randomly generate 25 instances, numbered from 26 to 50, to create the five cases of 80, 90, 100, 110, and 120% request density, with a single source location and a single destination location for each transportation request. The time windows for the requests start at a random moment in the first week and end at a random moment in the second week. Next, we

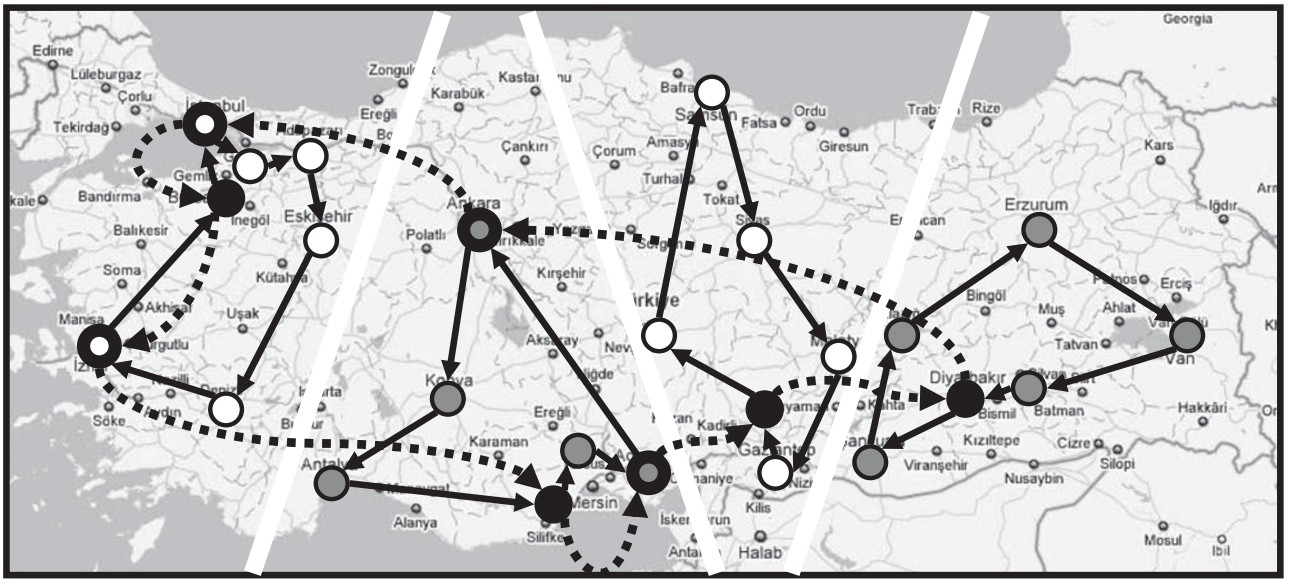

Figure 5 Routes of the couriers. Routes of couriers 1–4 are regional couriers that cover different geographical regions separated by the white lines. Route of courier 5, the cross-regional courier, is represented by dotted and curved lines.

Table 3 Stops of the couriers. Couriers 1–4 are executed every week. Courier 5 is executed biweekly

Stops	*Courier* 1	*Courier* 2	*Courier* 3	*Courier* 4	*Courier* 5
1	Bursa	Mersin	Kahramanmaraş	Diyarbakır	Bursa
2	Istanbul	Tarsus	Kayseri	Şanlıurfa	Izmir
3	Gebze	Adana	Samsun	Elazığ	Mersin
4	Adapazarı	Ankara	Sivas	Erzurum	Adana
5	Eskişehir	Konya	Malatya	Van	Kahramanmaraş
6	Denizli	Antalya	Gaziantep	Batman	Diyarbakır
7	Izmir	Mersin	Kahramanmaraş	Diyarbakır	Ankara
8	Bursa	—	—	—	Istanbul
9	—	—	—	—	Bursa

generate 25 similar instances, numbered from 51 to 75, with all requests having the same time window from the beginning of the planning period to the end. Notice that in this last setting, every request has at least one feasible path from its source to its destination. The results of these experiments are given in Table 4.

The average CPU time for model P2 is 0.076 sec for instances 26–50, and 36.5 sec for instances 51–75. The largest two CPU times are approximately 5 and 2 min (instances 75 and 72). All other instances have CPU times of less than 2 min. No branching was necessary in 24 instances of the first set and in one instance of the second set. The number of branch and cut nodes in the remaining 25 instances ranges from 1 to 4075 and is less than 1000 in all but three of these instances. The initial optimality gap is usually less than 1%, and is 0.29% on the average. The problems get harder to solve with increasing demand density. We conclude from these tests that Model P2 is effectively solved in low CPU time despite the fact that the number of zero/one variables is quite large (between 4000 and 5000 for different request density values).

We use the same data set to test the performance of model P3. The insertion of additional arcs to allow skipping of stops makes model P3 much denser in the number of arcs than model P2. The number of arcs in the time extended courier graph of model P2 is approximately squared in model P3. This causes model P3 to have in the order of 100 000 to 150 000 more binary variables than model P2. The results of these experiments are given in Table 5. The CPU times for the instances 26–50 increase about 100–200 times and range now from 6 sec to about 18 sec. This is still a very low CPU time for problems of this size with more than 100 000 binary variables. No branching is necessary for 23 of the 25 instances in this set while there are four and 24 branch and cut nodes in the remaining two instances. The instances 51–75, however, are considerably harder to solve now than in the case of model P2. The average number of branch-and-cut nodes is about five

Table 4 Performance of model P2

Instance	*Number of requests*	*B&C nodes*	*Initial optimality gap* (%)	*CPU time* (*sec*)	*Instance*	*Number of requests*	*B&C nodes*	*Initial optimality gap* (%)	*CPU time* (*sec*)
26	720	0	0.00	0.06	51	720	182	0.03	5.63
27	720	0	0.00	0.05	52	720	210	0.03	8.65
28	720	0	0.00	0.05	53	720	87	0.04	3.84
29	720	0	0.00	0.07	54	720	112	0.03	4.51
30	720	0	0.00	0.06	55	720	364	0.03	18.89
31	810	0	0.00	0.07	56	810	225	0.03	8.15
32	810	0	0.00	0.06	57	810	382	1.50	10.28
33	810	0	0.01	0.08	58	810	284	1.42	11.60
34	810	0	0.00	0.07	59	810	145	1.45	6.54
35	810	0	0.00	0.08	60	810	163	0.02	6.78
36	900	0	0.00	0.09	61	900	270	0.81	16.47
37	900	0	0.00	0.07	62	900	0	0.01	1.22
38	900	0	0.00	0.08	63	900	785	0.03	63.70
39	900	0	0.00	0.08	64	900	823	1.29	55.29
40	900	0	0.00	0.08	65	900	507	0.02	24.77
41	990	0	0.00	0.09	66	990	1117	1.11	71.33
42	990	0	0.00	0.07	67	990	146	0.66	5.91
43	990	0	0.00	0.09	68	990	930	1.00	63.20
44	990	0	0.00	0.08	69	990	190	0.36	9.94
45	990	0	0.00	0.08	70	990	277	0.36	16.30
46	1080	0	0.00	0.09	71	1080	366	1.01	17.54
47	1080	0	0.00	0.10	72	1080	1616	1.23	130.74
48	1080	0	0.00	0.09	73	1080	500	1.44	30.76
49	1080	0	0.00	0.09	74	1080	342	0.45	18.74
50	1080	1	0.01	0.09	75	1080	4075	0.06	302.03

Table 5 Performance of model P3

Instance	*Number of requests*	*B&C nodes*	*Initial optimality gap (%)*	*CPU time (sec)*	*Instance*	*Number of requests*	*B&C nodes*	*Initial optimality gap (%)*	*CPU time (sec)*
26	720	0	0.00	6.83	51	720	691	0.03	36.29
27	720	0	0.00	6.41	52	720	2250	0.03	78.70
28	720	0	0.00	6.44	53	720	748	0.03	35.95
29	720	0	0.00	7.03	54	720	313	0.03	33.55
30	720	0	0.00	6.77	55	720	1180	0.03	79.12
31	810	0	0.00	9.55	56	810	2064	0.03	112.55
32	810	0	0.00	9.04	57	810	2379	1.53	128.08
33	810	0	0.00	8.84	58	810	3877	1.44	723.79
34	810	0	0.00	9.91	59	810	290	1.48	37.91
35	810	4	0.00	8.91	60	810	2178	0.02	178.97
36	900	0	0.00	11.30	61	900	340	0.82	42.80
37	900	0	0.00	9.67	62	900	192	0.01	29.58
38	900	0	0.00	9.71	63	900	3668	0.02	787.82
39	900	0	0.00	10.49	64	900	5669	1.30	723.05
40	900	0	0.00	11.19	65	900	5309	0.02	702.78
41	990	0	0.00	13.82	66	990	632	1.12	44.36
42	990	0	0.00	11.76	67	990	373	0.67	50.88
43	990	0	0.00	10.47	68	990	12 546	1.01	2660.46
44	990	0	0.00	12.01	69	990	309	0.36	56.58
45	990	0	0.00	10.60	70	990	1541	0.36	69.49
46	1080	0	0.00	12.91	71	1080	4763	1.05	136.87
47	1080	0	0.00	14.09	72	1080	11 617	1.24	3732.24
48	1080	0	0.00	11.92	73	1080	3133	1.45	482.64
49	1080	0	0.00	13.68	74	1080	963	0.46	76.02
50	1080	24	0.00	18.11	75	1080	22 819	0.06	19732.35

times that of model P2. The average CPU time is 1230.9 sec (20.5 min) and the maximum CPU time is about 6 h (instance 75). Even though the average CPU time is about 20.5 min, there are only three instances (68, 72, and 75) that require a CPU time of more than 15 min. In all of these three instances, the optimality gap is less than or equal to 0.01% after 15 min. Noting that instances 51–75 are constructed to represent the worst case in terms of planning complexity, we can conclude that our models can be used in real world settings.

7.3. Summary of the computational experiments

Our computational experiments described in detail in the two previous subsections have shown that large scale instances (2-week planning period, 25 locations, nine couriers, 58 legs, 700–1100 transportation requests) of the original problem can be solved within 5 min of computing time on a desktop computer, using a general purpose commercial solver. The initial optimality gaps for models P1 and P2 are usually less than 1%, whereas it can increase up to 1.5% for P3. The instances involving random time intervals (instances 26–50) for transportation requests seldom require any branching, and are solved within a few seconds. Even for instances where all requests have the whole planning period as their time window (instances 51–75), the computation time requirement is no more than 1 CPU hour, except for one pathological instance that required 6 h.

8. Conclusion

We have posed and solved a real world delivery problem that requires a priority-based allocation of many-to-many transportation requests to a fleet of transportation assets that operate on fixed routes and at fixed time tables. The problem is new to the literature. We have proposed multi-commodity flow models based on a time extended network that we construct from the routes and schedules of the transportation assets under consideration. The main model is solved in low CPU time for problem sizes encountered in the real world while the model with an option to skip stops is solved in relatively higher CPU time due to the substantially increased number of binary variables. Our main contributions are to develop an effective formulation scheme for a complicated large-scale real world problem and to demonstrate that such problems are solvable via commercial general purpose solvers through meticulous modelling.

The option of skipping stops is more meaningful when the request density for deliveries is relatively low than when it is relatively high. It is advisable to use the main models P1 or P2 in case of high demand since courier stops are rarely skipped when the request density is high.

The priority-based construction of the objective function coefficients leads in general to a wide range of values for these coefficients while the percent differences in objective values are relatively small. This indicates that the number of accepted items is relatively stable with respect to possible shifts in delivery routes of items.

One possible venue for future research may be the variant of the CPP for which courier routes and schedules should be designed based on the transportation requests. This variant may be of further interest to the practitioners in the commercial sector, as it focuses more on the cost. However, modelling it with the methods described in this study will require the discretization of the time element, and will result in a much higher number of nodes, arcs, and consequently number of variables and constraints. Solution methods based on path-based models utilizing column generation techniques seems more promising for this variant.

Acknowledgements—Thanks are due to the referees for their valuable comments.

References

Akgün I and Tansel B (2007). Optimization of transportation requirements in the deployment of military units. *Comput Opns Res* **34**: 1158–1176.

Baker SF, Morton DP, Rosenthal RE and Williams LM (1999). *Optimizing strategic airlift*. Technical Report NPS-OR-99-004, Naval Postgraduate School, Monterey, CA.

Baker SF, Morton DP, Rosenthal RE and Williams LM (2002). Optimizing military airlift. *Opns Res* **50**: 582–602.

Baveja A and Srinivasan A (2000). Approximation algorithms for disjoint paths and related routing and packing problems. *Math Opns Res* **25**: 255–280.

Bazaraa MS, Jarvis JJ and Sherali HD (1990). *Linear Programming and Network Flows*. Wiley: New York.

Berbeglia G, Cordeau J-F, Gribkovskaia I and Laporte G (2007). Static pickup and delivery problems: A classification scheme and survey. *TOP* **15**: 1–13.

Cormen TH, Leiserson CE and Rivest RL (2000). *Introduction to Algorithms*. MIT Press: Cambridge, MA.

Ford LR and Fulkerson DR (1956). Maximal flow through a network. *Can J Math* **8**: 399–404.

Fréville A (2004). The multidimensional 0-1 knapsack problem: An overview. *Eur J Opl Res* **155**: 1–21.

Fulkerson DR and Dantzig GB (1955). Computation of maximal flow in networks. *Nav Res Log Q* **2**: 277–283.

Garey MR and Johnson DS (1979). *Computers and Intractability*. Bell Telephone Laboratories: Murray Hill, NJ.

Grinold RC (1968). A multicommodity max-flow algorithm. *Opns Res* **16**: 1234–1238.

Grinold RC (1969). A note on multicommodity max-flow. *Opns Res* **17**: 755.

Guruswami V, Khanna S, Rajaraman R, Shepherd B and Yannakakis M (2003). Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *J Comput Syst Sci* **67**: 473–496.

Karp RM (1975). On the computational complexity of combinatorial problems. *Networks* **5**: 45–68.

Kleinberg J (1996). Approximation algorithms for disjoint paths problems. PhD thesis, Massachusetts Institute of Technology.

Kolliopoulos SG and Stein C (2002). Approximation algorithms for single-source unsplittable flow. *SIAM J Computing* **31**: 919–946.

Kolman P and Scheideler C (2006). Improved bounds for the unsplittable flow problem. *J Algorithm* **61**: 20–44.

Sherali HD (1982). Equivalent weights for lexicographic multi-objective programs: Characterizations and computations. *Eur J Opl Res* **11**: 367–379.

Received April 2010;
accepted November 2010